AR-006-138

DSTO-TR-0151

A Fiber Distributed Data Interface (FDDI) Network Analyser

Alan Allwright, Reg Driver and Alan Wood

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

A Fiber Distributed Data Interface (FDDI) Network Analyser

Alan Allwright, Reg Driver, Alan Wood

Information Technology Division
Electronics and Surveillance Research Laboratory

DSTO-TR-0151

ABSTRACT

The trend towards distributed Command and Control systems in Naval platforms necessitates the provision of Local Area Network performance measurement tools and techniques. A special purpose network analyser has been developed within ITD to measure the performance of a Fiber Distributed Data Interface (FDDI) network. This paper describes the architecture and operation of the network analyser.

This work has been conducted to support the Directorate of Naval Combat Systems Engineering in their Local Area Network and distributed systems analysis.

APPROVED FOR PUBLIC RELEASE

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury. South Australia, Australia, 5108

Telephone: (08) 259 7085
Fax: (08) 259 5980

AR-008-138
February 1995

APPROVED FOR PUBLIC RELEASE

A Fiber Distributed Data Interface (FDDI) Network Analyser

EXECUTIVE SUMMARY

To support the selection, design, development, maintenance and upgrade of local area networks in current and planned distributed Command and Control systems detailed knowledge of the behaviour of candidate local area networks needs to be known. The development of the analyser described in this report was conducted to support the Directorate of Naval Combat Systems Engineering in their network performance analysis activities.

The network analyser described in this report supports the process of performance data collection and real time monitoring of a Fiber Distributed Data Interface local area network.

This report provides technical details of the operation and implementation of the network analyser. The description covers the design and set up of the hardware and software necessary to operate the analyser and collect performance data in real time.

A second report "Test-Bed Performance Analysis of the Fiber Distributed Data Interface" (Report No. DSTO-TR-0150) provides a description of validation and trial experiments conducted with the network analyser.

Authors

Alan M. Allwright
Information Technology Division

Alan Allwright graduated in Mathematics and Computing from the South Australian Institute of Technology in 1988. Between 1990 and 1994 *Alan worked on a Masters degree in Computing. His thesis titled "Performance Analysis of Distributed Databases for Combat Systems" was submitted in 1995. Alan now works on computer simulation and System Engineering related tasks within the C3ISE group of ITD.*

Reg. Driver
Information Technology Division

Reg retired as a SITO in 1993.

Alan Wood
Information Technology Division

Alan is an ITO in C3ISE group. He is studying an Associate Diploma in Information Systems at the Torrens Valley Institute of TAFE. Alan is responsible for network management, computer support, and Systems Engineering related tasks within C3ISE.

CONTENTS

Figures

Tables

Appendices

ABBREVIATIONS

AMD	Advanced Micro Devices
ANSI	American National Standards Institute
CMT	Connection Management
CS	Completion Of Service
DMA	Direct Memory Access
DOC	Digital Optical Converter
DPC	Data Path Controller
DPTBS	Distributed Processing Test-Bed System
EAR	End of Receive FIFO
EDS	ENDEC Data Separator
ENDEC	Encoder Decoder
FDDI	Fiber Distributed Data Interface
FIFO	First In, First Out
IAT	Inter-Arrival Times
ISO	International Standards Organisation
ITD	Information Technology Division
LAN	Local Area Network
MAC	Media Access Controller
Mbps	Megabits per second
ODC	Optical Digital Converter
PC	Personal Computer
RAM	Random Access Memory
RBC	RAM Buffer Controller
RPR	Read Pointer for Receive FIFO
RPXA	Read Pointer Asynchronous frames
RPXS	Read Pointer Synchronous frames
RT	Request Time
SAR	Start Address Receive FIFO
SAS	Single Attachment Station
ST	Start Time
T_OPR	Operative TTRT
TRT	Token Rotation Time
TTRT	Target Token Rotation Time
WPR	Write Pointer Receive FIFO
WPXA	Write Pointer Asynchronous frames
WPXS	Write Pointer Synchronous frames
kb	Kilobyte
km	Kilometer
ms	milliseconds
ns	nanoseconds
us	microseconds

1 Introduction

The Distributed Processing task (NAV87/226.3) has conducted an investigation into the operation and performance of the FDDI protocol. This has prompted the development of a Personal Computer (PC) based network analyser. A number of technologies suitable for the distributed management of data on naval platforms (Ref.1) were assessed. The Fiber Distributed Data Interface (FDDI) Local Area Network (LAN) protocol is a standardised (ANSI, ISO) high speed fault tolerant token ring protocol. Fault tolerance and high bandwidth make FDDI and its derivative FDDI-II excellent candidates for the LANs on future naval platforms.

The study involved the use of a simulation model (Ref. 2) which allows the user to specify the network configuration and traffic demands typical of a combat system. The analyser was developed to collect data to validate this simulation model. It is also useful for other applications where the frame transmission delays through the FDDI Media Access Controller (MAC) are required. The analyser has been designed to give the operator considerable flexibility in specifying frame arrival rates, frame types, and frame priorities at each node. It is also possible to set network parameters such as the operational timer (T_OPR)(Ref. 4) and the ring latency.

An overview of the network analyser's architecture is provided in Section 2. The components and operation of each of the transmit nodes is presented in Sections 3 and 4. Sections 5 and 6 detail the construction and operation of the delay units and data logger. Section 7 discusses the general purpose software used for pre- and post-processing of network analyser data. The limitations and the operation of the network analyser are reviewed in Sections 8 and 9. Section 10 provides some typical results.

2 Network Analyser Architecture

Information Technology Division (ITD) has produced a Distributed Processing Test-Bed System (DPTBS (Ref. 3)). The DPTBS has the following objectives :

- To develop distributed processing applications.
- To implement distributed database management protocols for evaluation and demonstration.
- To investigate and evaluate the FDDI LAN protocol.
- To provide a platform for validating distributed database simulation models.

The test-bed comprises several IBM-compatible 80386, 20 Mhz Personal Computers (PCs). Each PC is connected as a node to an FDDI network using an Advanced Micro Devices (AMD) FDDI "Fast Card" (Ref. 8). The FDDI cards provide a Media Access Controller (MAC), a Data Path Controller (DPC), a RAM Buffer Controller (RBC) and 128 Kb of buffer memory; each of these components is discussed in Section 3. A software package, PDEMO, supplied with the FDDI cards (Ref. 10), is used to initialise the FDDI hardware and to perform the network connect for each node.

The DPTBS forms the basis for the network analyser. Each network analyser node is formed by the addition of a timer / interrupt card, a delay unit (see Section 5) and an application software package (see Section 3), for generating transmit requests from each DPTBS node. The timer/interrupt card on each node synchronises the clocks on each transmitting node with a master clock in the data logger. A data logger (IBM compatible AT PC, see Section 6) is connected to the nodes via the data bus (Ref. 8) on one of the FDDI cards. Figure 1 depicts a 3 node DPTBS configuration interfaced to the data logger.

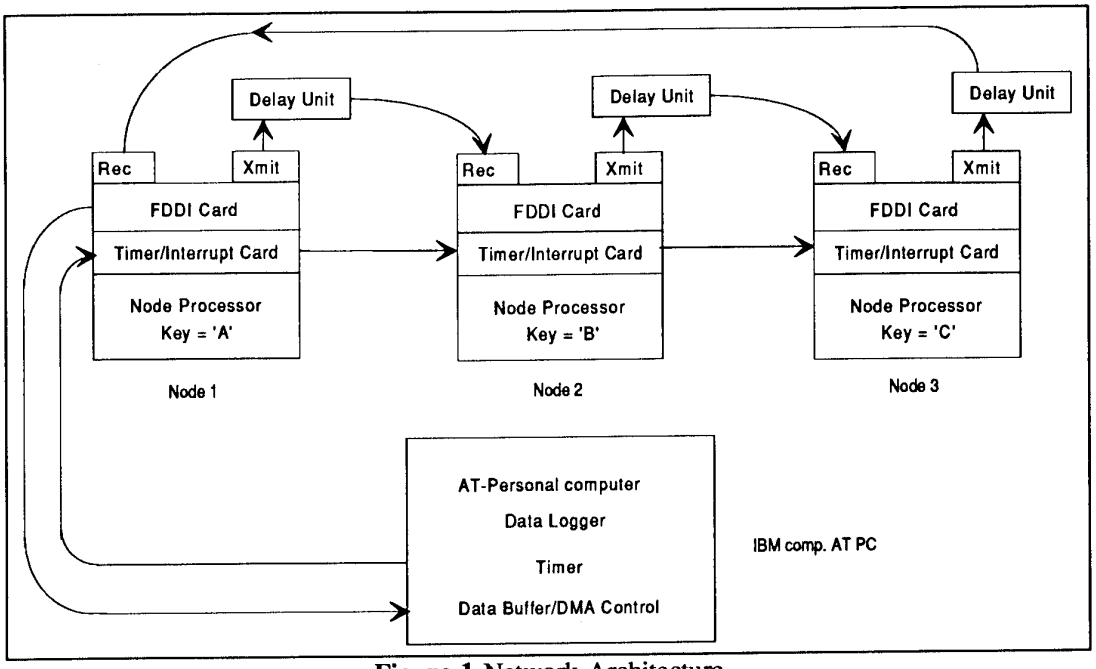

Figure 1 Network Architecture

A typical FDDI network uses a Connection Management (CMT) protocol (Ref. 5) to form a dual attachment ring or tree topology. In this case the network configuration used bypasses CMT and forms a (non CMT compliant) ring of Single Attachment Stations (SAS). This was done due to limited resources and does not affect the performance measurements being made.

The objective of the network analyser is to determine the frame transmission delays through Media Access Controller (MAC) at each node. In order to achieve this objective, the transmit software on the three transmitter nodes (see Section 3) generate requests independently at pre-specified times. The requests are assigned a node key which is unique to the node originating the request. Node keys (Key = 'A', 'B', 'C') are loaded into the 'frames' data fields to allow the data logger to determine the source of the frames being logged. These keys are used during the analysis of the data to match data logger times with the frame transmit times. As each frame is transferred through node 1, it is logged via the data bus on the FDDI Cards. The data logger saves the time the frame is logged and its key for later processing; the data in the frames are discarded. Frames logged in this manner also experience delays in their transmission and propagation on the network.

These delays are attributable to processing at the Physical Media Dependent (PMD) (Ref. 6) and PHYsical (Ref. 7) levels in the FDDI protocol suite. These delays are generally small in comparison to the MAC delays. Consequently the delays experienced after the MAC processing will not significantly bias the results.

3 Transmitter Nodes

The transmit program transmits frames from the transmitter node processor onto the network. The main difficulty with the transmitter is its inability to load the network to 100 Mbps with a single node. With PCs, the maximum continuous data transfer rate between the PC main memory and the FAST Card buffer memory, is less than 30 Mbps (Ref. 9).

Since the experiments are not concerned with the data content of the frames, it is possible to pre-load the frames to be transmitted into the FDDI card buffer memory and loop back the memory pointers. In this way, the same frames are used over and over again without the necessity of loading new frames into the buffer between transmit requests. Memory pointers are used by the node processor, Data Path Controller (DPC) and Ram Buffer Controller (RBC) to monitor and control the locations in buffer memory where data is to be read and written. The pointer types are presented in Table 1. By removing the necessity for the node processor to manage any frame data, all memory pointer and data transfer operations can be handled automatically by the FDDI Cards, and a single transmit enable command can be used to transmit up to 720,000 bits (20 frames of 4500 bytes). Since transmit enable requests can be made every 256 microseconds, each transmitter node is now capable of handling requests at 2,812 Mbps, which is well in excess of the required 100 Mbps.

3.1 Transmit Parameters

The main processes in the transmit program are indicated in Figure 2. The addresses used in the transmit program for calls to the FDDI card registers are listed in Appendix II. The experimental parameters, are entered at the start of each run.

The user is prompted for each of the relevant experimental parameters in the following order :

- Batch Length; The batch length is the number of frames to be transmitted with a single transmit request, with a limit of 20 frames.

- Frame Length; The frame length is the length of the frame to be transmitted, minus the frame source and destination addresses, frame control and end delimeter (20 Bytes). The frame length may be up to 4480 bytes.

- Message Type (Synchronous or Asynchronous); If the message type is asynchronous the user is prompted for the asynchronous priority. The asynchronous priority is set between 2560 ns (high priority) and T_OPR (low priority).

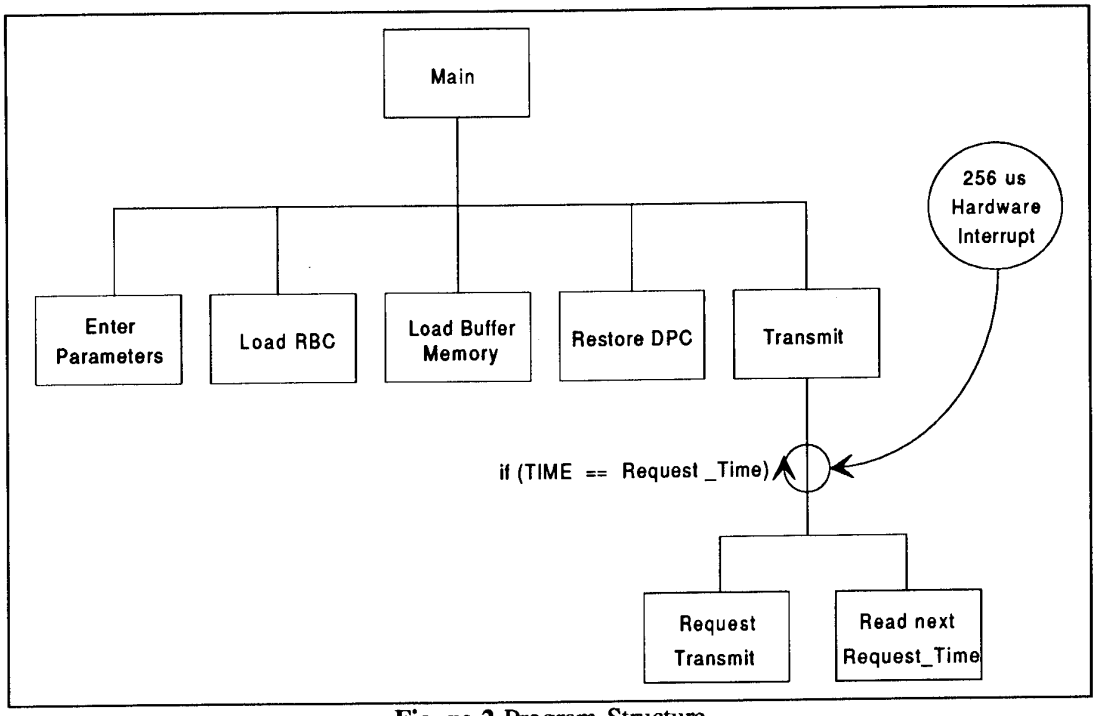

Figure 2 Program Structure

The transmit program uses the batch length and frame length to initialise the frames in buffer memory. The frame type can then be selected as either synchronous or asynchronous.

3.2 Buffer Memory

FDDI Cards are equipped with 128 Kb of buffer memory organised as 32 Kb X 32 bits. Once the RBC is initialised the frame data is loaded into Buffer Memory. Two batches of frames are loaded into buffer memory as indicated in Table 1. The start address for the data is set during initialisation at 0x100. The data for the batches is loaded into the buffer memory and the frame start address pointer for the last frame in the second batch is set to point back to the start of the first batch (0x100).

The load buffer memory procedure (Figure 2) requires the following parameters - Source Address, Destination Address, Frame Data. The Source Address is read from the FDDI card on the transmitting node. The Destination Address and Frame Data were determined in advance and are written automatically by the transmit software. This data is then loaded into the buffer memory at the addresses indicated by the write pointer (WPR).

Table 1 Buffer Memory Pointer Assignment

Receive FIFO	0x0	
Asynchronous Transmit FIFO	0x100	Start of Batch 1
		Batch 1 Frames
	0x2cfe	Start of Batch 2
		Batch 2 Frames
	0x58fa	Loop back to 0x100
Synchronous Transmit FIFO	0x7000 - 0x7fff	

3.3 RAM Buffer Controller (RBC)

The RBC is supplied by AMD as part of the FDDI chip set. It manages buffer addresses for the receive and transmit requests. A table of pointers (see Table 2) stored in memory in the transmit node processor is used by the node processor transmit software, the RBC and the DPC to manage data received and to be transmitted. The node processor uses the pointer RPR for reading data from the buffer memory and WPR for writing to the buffer memory. The DPC uses the RBC to move data between the fibre media and the buffer memory. The RBC is initialised by writing the calculated start and end addresses for the receive, synchronous, and asynchronous transmit buffers. Table 2 shows the initial RBC and RAM buffer address assignments.

Table 2 RBC and Buffer Memory Address Assignments

Pointer	Asynch	Synch	Description
RPR	0x0	0x0	Read pointer for Receive FIFO
WPR	0x0	0x0	Write pointer for Receive FIFO
SAR	0x0	0x0	Start address for receive FIFO
EAR	0x0	0x0	End address for receive FIFO
WPXA	0x100	0x50	Write pointer for Asynchronous frames
RPXA	0x100	0x50	Read pointer for asynchronous frames
WPXS	0x7000	0x100	Write pointer for synchronous frames
RPXS	0x7000	0x100	Read pointer for synchronous frames

3.4 Data Path Controller (DPC)

The DPC is responsible for transferring data received from the fibre media to buffer memory and for transferring data to be transmitted from buffer memory to the fibre media. The DPC utilises the address pointers managed by the RBC to determine the location to transfer data either into or out of buffer memory.

The DPC must be reset after the RBC registers have been set. This involves restoring the pointers (WPXS, WPXA) in the RBC, and resetting the transmit enable locks in the DPC.

3.5 Media Access Controller

The Fiber Optic Ring Media Access Controller (FORMAC) controls access to the ring by performing the FDDI MAC protocol (Ref. 4). Media access to the ring is governed by a timed token rotation protocol. A transmitting node can only transmit synchronous frames onto the network when it captures the token. A transmitting node can only transmit asynchronous frames when it has captured the token and there is sufficient unutilised bandwidth on the current token rotation. The amount of unutilised bandwidth on each token rotation is calculated using the Token Rotation Timer (TRT) and Target Token Rotation Time (TTRT) at each transmitting node.

The FORMAC determines the type of frame to send (synchronous or asynchronous), reads the data from the buffer memory, constructs the frame and sends it to the transmit hardware (Ref. 10). The FORMAC is also responsible for the maintenance of the token.

The transmit program only interacts with the FORMAC by setting the asynchronous priority level (Appendix II) which the FORMAC uses to determine whether the frame can be transmitted on the current token capture (Ref. 4).

4 Transmitter Operation

Once all the components discussed in Section 3 have been initialised, the FDDI cards are ready to transmit the frames loaded in buffer memory. The transmit procedure initially reads the request times as text from the request file (Section 6) and writes them to an array as long (4 bytes) words. Storing the request times in advance in a main memory array, rather than calculating them on the fly, or retrieving them from hard disk, significantly decreases the processing overhead required in making transmit requests.

The following pseudo code shows how the transmit procedure maintains the current time and schedules transmit requests;

```
Read_Count = Number_of_Requests;
Get Request_Time;

While (Request_Count < Read_Count)
     {

     Wait for Interrupt;

     Increment Current_Time;

     If (Current_Time >= Request_Time)
          {
          Enable_Transmit;
          Get next Request_Time;
          }
     }
```

The Number_of_Requests is set equal to the number of request times read into the array. The first Request_Time is read from the request time array. The main loop of the code then waits for the timer interrupt before testing whether or not it is time to request a transmit. If it is time to request a transmit, the enable transmit command is issued to the DPC (Appendix II), the next request time is read from the request time array, and the loop then returns to wait on the timer interrupt. If it is not yet time to make the transmit request the program loops back to wait on the timer interrupt.

After all the transmit requests have been processed the program waits for the operator to terminate the program.

5 Delay Units

The delay units were developed to control the ring latency. By introducing the delay units it is possible to emulate the effects of networks with more nodes and longer fibre optic cable lengths than that available with the DPTBS.

The delay unit (Figure 3), consists of commercial FDDI ring interface circuits and a variable length First-In-First-Out (FIFO) memory (developed in ITD) to produce the required delay. The delay is achieved by varying the length of the FIFO buffer.

The unit provides switch selectable delays from 980 ns to over 2 ms in incremental steps of 320 ns. This delay, based on 5085 ns/km (Ref. 4), represents between 0.19 km and 392 km of fibre optic cable.

5.1 Delay Unit Operation

The delay units are connected to the fibre optic cable (refer Figure 1). Data received by the delay units is placed in a FIFO queue (5.1.2) where it is delayed for a predefined period of time. Once the data has been delayed it is re-transmitted on the network.

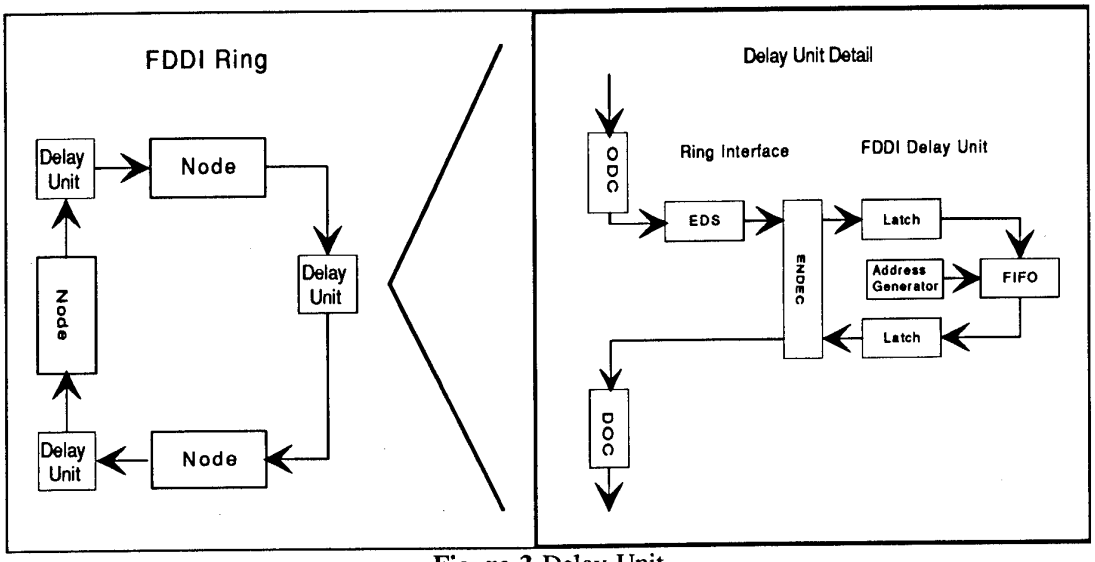

Figure 3 Delay Unit

5.1.1 Ring interface

The serial optical information from the ring is converted to serial digital data by the optical-to-digital converter (ODC). This information is then passed to the FDDI ENcoder DECoder (ENDEC) data separator (EDS), where the clock and data are recovered from the FDDI data stream.

The (clock and data) information is passed to the ENDEC for conversion into parallel words (8 bits per word). The information is then transferred to the delay circuitry for storage.

5.1.2 Delay FIFO

Since the information rate from the ENDEC is too fast for the FIFO memory delay circuits, the 8 bit data are fed into a 16-bit latch before being passed to the memory, thereby halving the memory data rate.

When data are read from the memory the reverse process takes place. The 16 data bits are read from the memory into a latch as 4 symbols of 4 bits. The latch passes the symbols (in pairs) to the ENDEC for conversion into serial data. The data is then sent to the digital-to-optical converter (DOC) for return to the FDDI ring.

A binary down counter is used to generate memory addresses. This counter is preset to the required starting address (count) by switches and clocked until address zero is reached. At this point the counter is automatically set back to the starting address and the process is repeated. When the address generator selects a FIFO memory location, the contents of that location are written to the output latch for passing to the ring (via ENDEC and DOC).

New data from the ring are then accepted from the input latches and written into the same memory location. The address generator then selects the next memory location and the process is repeated. The required delay is achieved by controlling the number of memory locations in use.

The minimum delay is selected by setting the switches to zero. With this setting only one memory location is used and the data are written into that memory location and immediately read out again.

There is a minimum delay introduced by the integrated circuits and the clocking of data. The insertion of the delay unit in the ring introduces a minimum delay of 980 ns which must be added to the preset switch delay.

The incremental delay (320 ns) is determined by the address generator clock rate. Thus a setting of two on the switches would produce a delay of ;

Delay = 980 ns + (2 * 320 ns) = 1.62 μs.

6 Data Logger

The data logger was developed to capture and log events on the FDDI network. The events are the reception of frames by the transmitter node connected to the data logger. When an event occurs, the key (see Section 2) and the time the event occurred are saved by the data logger.

The data logger consists of the following main elements (Figure 4) :

- FDDI-PC interface: The FDDI-PC interface extracts the keys from the FDDI card and sends this information to the event latches in the data logger.
- Event latch: The event latches record the events for later reading.
- FIFO buffer: The FIFO stores the information so the PC can retrieve it.
- PC interface: The PC interface allows the PC to read information from the FIFO.
- DMA Software: The software sets up the PC for DMA transfer and handles the transfer of data from buffer memory in the data logger to the main memory in the data logging node.

Each of the transmitter nodes on the network is fitted with a circuit that generates an interrupt to the data logger when signalled from the master timer. The timer ticks are distributed by the master timer, located on node 1, to synchronise the ticks on the three nodes. When a preset number of ticks has occurred, a request for transmission is made to the FDDI card.

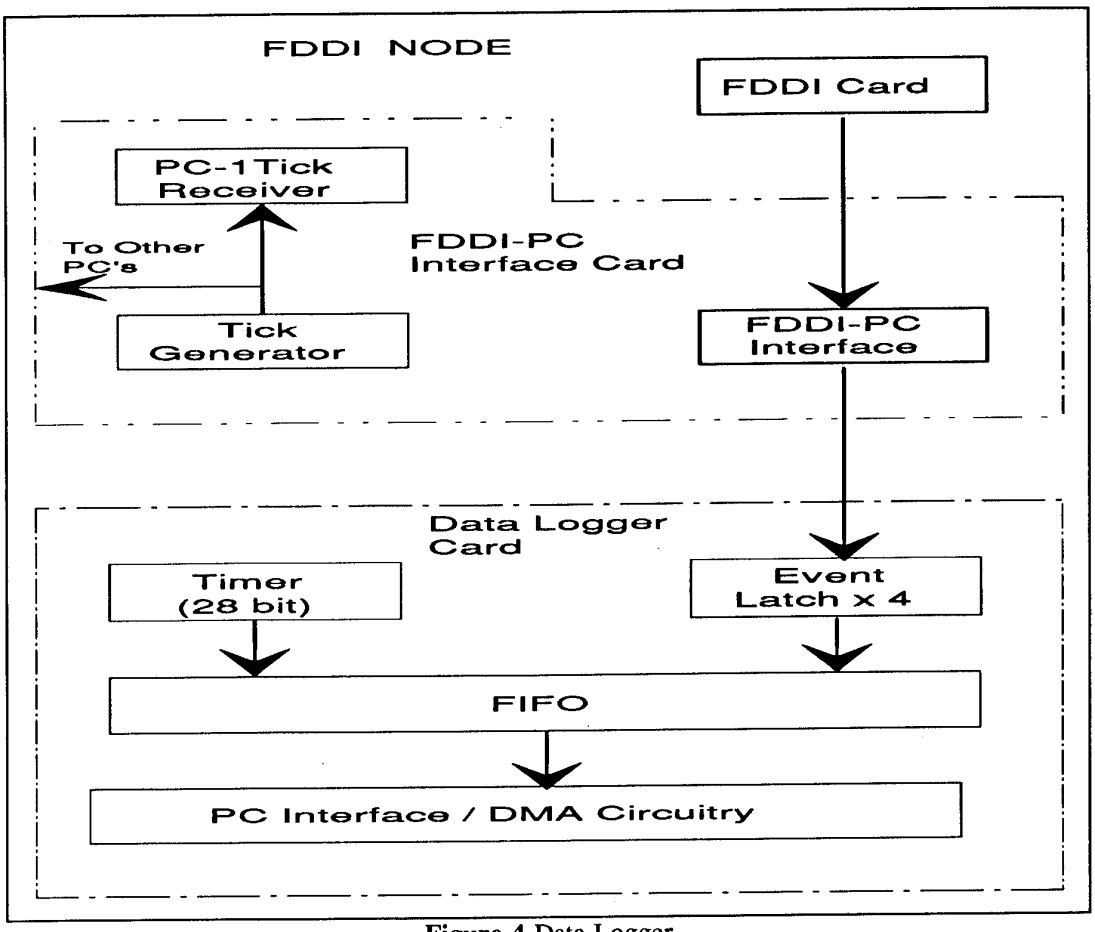

Figure 4 Data Logger

A circuit in the FDDI-PC interface card detects the transmitting PC's station key and sends that information to the data logger which then logs that event and the time that it occurred. Up to four events can be logged simultaneously.

A 28 bit counter in the logger is used to keep track of the elapsed time. The timer has a resolution of 500 ns. Another 4 bits are used to record the events. The time and events are combined into a 32 bit word. This word is then stored in a data file for later analysis (see Section 7.2).

6.1 Data Logger Operation

When the first event occurs, the timer is reset to zero and the rest of the system is enabled. All times logged are relative to this point.

The latches act as temporary event stores. When the data are read from the latches, the latches are cleared. The elapsed time and the events that triggered the action are moved to the FIFO. Since the FIFO is only 8 bits wide and we want to store a 32 bit word, this has to be done in 4 by 8 bit transfers.

When there is data in the FIFO, a signal to the PC is raised. The signal initiates a DMA transfer. The transfer continues until all the data is moved from the FIFO into a RAM disk in the data logging PC. Transfers from the FIFO to the data logger can be made while events are being logged.

If the FIFO fills up, there is no mechanism to halt the experiment and data will be lost. The operator is informed when buffers are lost and the experiment can be restarted.

Data transferred to the RAM disk is stored in 4 bytes - the first 4 bits hold the station key and the remaining 28 bits hold time-stamp information.

7 Generator and Analysis Software

The generator and analysis software was designed for the pre- and post-processing of network analyser data. Two packages were developed. These are request generators (Section 7.1) which produce files containing the inter-arrival times for transmit requests, and the analysis program (Section 7.2) which processes the data produced by the network analyser to calculate the average MAC delays and message loss probabilities (see Section 10). Both of these packages are used separately from the network analyser.

7.1 Request Generators

The request generators are used before an experimental run to produce the request files for each of the transmitting nodes. There are two request programs. The first generates request times for fixed Inter-Arrival Times (IAT). The user is prompted for the run length of the experiment and the IAT for arrivals. The program then generates a sequence of request times with the given IAT for the period of the run. The IAT's generated by this program are usually used for synchronous requests.

The second random arrival generator creates a file of arrivals with uniformly distributed arrival times. The user enters the number of arrivals, the expected IAT of arrivals and a seed to initialise the random number generator. The program uses the Turbo C++ internal random number generator and produces IAT in the interval between one and two times the expected IAT (less 0.5 micro seconds). The IAT's generated by this program are usually used for asynchronous requests.

7.2 Analysis Program

The analysis program is used after an experimental run to analyse the data stored by the data logger. It uses the files produced by the arrival generators and the data logger to couple the request events to the service events. The user is prompted for the input data file names, batch size, the frame length and the end time for the experiment. The end time for the experiment is taken as the point when all synchronous requests have been serviced.

The number of synchronous and asynchronous requests are calculated in advance so that synchronous requests will complete before asynchronous requests. This allows asynchronous requests to completely fill the buffer in the data logger. When synchronous requests are being analysed the end time is set to zero to indicate all synchronous requests are to be processed.
When the synchronous requests have been processed the time of service of the last request is displayed. This last synchronous service time is then used for the end time for the processing asynchronous requests. The user may select an option to display a detailed listing of the matching between the arrival and results file, or display a summary of the results. The detailed listing shows each request, and its request time, matched to its corresponding service and the service time. A discussion of the detailed listings is given in Appendix I. The summary results for a typical run are discussed in Section 10.

The following pseudo-code shows how the algorithm matches request times to service times.

```
Read First_Request_Time;
First_Service_Time = First_Request_Time;

Read Second_Request_Time;
Second_Service_Time = Second_Request_Time;

Read Next_Request_Time;

While (not end of service times file)
     {
     If (time of completion of service for the
          First_Service_Time is later than the
          Next_Request_Time and the buffer is full)
          {
          disregard the request;
          Read Next_Request_Time;
          }
     else
          {
          First_Service_Time = Next_Request_Time;

          Read Next_Request_Time;
          Read Next_Service_Time;

          Second_Service_Time = First_Service_Time;
          Next_Service_Time = Second_Service_Time;
          }
     }
```

The program reads the request and service files in order and matches requests to services. Given that the FDDI cards have a buffer capacity of two requests (Ref. 9) the first and second request times are automatically assigned to the first and second service times. The program then simulates the operation of the FIFO buffer to couple the request and service times. The first service time relates to the service time at the head of the buffer, the second service time relates to the second element in the buffer. Request and service times are read sequentially, and the program must determine if the request was serviced. A count is maintained of the number of requests currently in the FIFO.

If the buffer is full and the current request time is earlier than the service time for the service at the head of the FIFO, the request is lost. If the request was serviced, the request time is matched to the service time, the second service time is moved to the head of the FIFO and the next service time is read and saved as the second element in the FIFO.

This program combines the arrival and service files in the way shown in Appendix I. It also calculates a number of statistics.

There are a number of ways to analyse this data and as such the statistical calculations would generally be left to an analyst. To provide an indication of the available summaries, Section 10 discusses some typical results.

8 Limitations

The network analyser has a number of limitations imposed by the hardware and software. These are discussed below:

- Each PC has the capacity to buffer up to 10,000 transmit requests.

- Each request is stored in an array of unsigned long integers. The combination of the size of an unsigned long integer (2^{32}-1) and the transmit clock granularity (256 μs) limits the maximum run length of any experiment. The maximum run length of any experiment is then;

  Run length = Maximum Time * Clock granularity
  = (2^{32}-1) * 256 μs = 1099511 seconds

- Each FDDI Card has a buffer capacity of 256 kb. The experiments require only two batches of 20 frames with 4480 bytes per frame. The maximum buffer requirement is:

  Buffer Requirement = batch count * batch size * frame length
  = 2 * 20 * 4480 = 179,200 bytes.

- FDDI Cards allow for a maximum of two synchronous and two asynchronous transmit requests to be buffered at any one time (Ref. 10).

- Each transmitting node uses a 256 μs clock. The timer/interrupt card allows other clock resolutions to be selected. The 256 μs resolution allows transmitting nodes to generate traffic up to the full bandwidth of the FDDI (100 Mbps) for frames 3200 bytes or longer. This resolution also provides for long experimental run times with a maximum of 11718 events occurring each second.

- The data logger can differentiate events with a resolution of 500ns. With bit lengths in FDDI being 10 ns, this resolution allows the data logger to differentiate between the smallest allowable FDDI frames (20 bytes).

- Logged data is stored in a RAM Disk for latter analysis. The size of the RAM Disk is limited only by available memory in the data logger. In our case 384 Kb was sufficient.

- Each Delay unit introduces a minimum delay of 980 μs. Delays up to 2 ms are switch selectable in steps of 320 ns.

9 Operation

The network is made operational by using the AMD PDEMO program (Ref. 8). By running the PDEMO program the operator is given the option to set a number of system parameters including network configuration and the TTRT.

The network is currently set up to run only the FDDI 'Class C' configuration, which allows the network to run with one transmitter and one receiver per node (Option 15). The "Class C" option bypasses connection management but has no effect on the performance of the operational network. The operative TTRT (T_OPR) is set to 4.0 ms as one of the MAC parameters according to the AMD "FAST Card" manual (Option 10) (Ref. 10). After the operational parameters for each node have been set, the operator exits the PDEMO program leaving the network operational (Option 199).

The transmit program on each node is then run and given its operational parameters (batch length, frame length, synchronous or asynchronous request type and priorities). For the example run described in this paper (Section 10), node 1 makes synchronous requests, node 2 makes asynchronous high priority requests and node 3 makes asynchronous low priority requests. The asynchronous priorities are set to high priority = 2560 ns and low priority = 3372000 ns. Synchronous requests are made in batches of 10 frames of 2688 bytes. Asynchronous requests are made for single frames of 4480 Bytes. Once the operational parameters have been entered on each node the program waits for a signal from the data logger to start the experiment.

The distribution of IATs for frames, at each node, is left up to the operator (currently fixed and random inter-arrival times are used). The IATs are stored in an ASCII text file (REQUEST.TXT) that is read into memory in each of the transmitting nodes. The transmit program reads the REQUEST file into an array which is accessed sequentially during the running of the transmit program. For the example run, node 1 uses fixed IATs of 3072 μs and nodes 2 and 3 use random IATs with a mean of 512 μs.

The data logger is initialised by using the data logger program (Section 5) on the data logger PC AT, which resets the data logger clock and DMA software. When all the transmitters and the data logger are ready the operator can start the experiment by enabling the master clock tick generator on node 1.

10 Analysis of Results

This section discusses the results for a single run of the network analyser. These results were produced using the analysis program described in Section 7.2. A detailed listing of the results for this run are discussed in Appendix I.

During the run, one node is generating synchronous requests at a rate of 70 Mbps. The other nodes are generating asynchronous requests at a rate of 70 Mbps. The net request rate is 210 Mbps.
The analysis program (Section 7.2) is used to analyse the results of the experimental run. The following parameters are entered when the program is initialised;

- Enter the data file : 70keyA.dat
- Enter the batch size : 1
- Enter the frame length : 2688
- Enter the last service time : 0

The data file name is the text file in to which all the results are written, in this case 70keyA.dat for "70" Mbps, synchronous (Key="A"). The batch size and frame length are discussed in Section 3. The last service time is entered as the time for the program to stop processing results. In this case all the requests are to be processed and this parameter is set to zero. For asynchronous requests the last service time is set to the value produced in the analysis of the synchronous file. Because the synchronous requests are designed to finish before the asynchronous requests, it is necessary to specify the last service time of synchronous requests to stop the asynchronous statistics becoming biased by the end conditions of the experiment. The following results were produced by the analysis program (Table 3). All times are in nanoseconds. In this case the data were provided by a trial run of the network analyser where the required synchronous throughput was 70 Mbps.

Table 3 Experimental Results

Last service time == 12292232

Total requests = 20000, Accepted requests = 20000

Delay max = 7182.000000, Delay min = 0.000000

Delay sum = 2360026.000000, Delay ssq = 4388019200.000000

Mean = 118.001297, Var = 205486.937500

95% Confidence Interval = 6.282510

99% Confidence Interval = 8.269834

buffer 0 prob = 0.816000

buffer 1 prob = 0.184000

buffer overflow prob = 0.000000

The last service time (12292232) is the time (number of 500 ns clock ticks) the last synchronous request was serviced. This value is used for the last service time parameter in the analysis of the asynchronous requests.

The total requests (20000) is the number of synchronous requests made during the run. The accepted requests (20000) is the number of requests that were serviced by the MAC, the number of rejected requests is the number of requests that were lost due to buffer overflow.

The maximum (7182.0) and minimum (0.0) delays are the maximum and minimum times the requests for synchronous transmission spent in the buffer before being processed by the MAC. The delay sum (2360026.0) and delay sum of squares (4388019200.0) are counts used to calculate the mean, variance and confidence interval statistics for the MAC delays. The mean is the mean delay (118.001297) for a requests waiting in the buffer, and the variance (205486.9375) is given for the mean. The confidence intervals are calculated assuming Students 't' distribution (Ref. 12) :

$$\bar{X}(n) \pm t_{n-1,1-\alpha/2} \sqrt{\frac{S^2(n)}{n}}$$

Where n is the sample size, $\bar{X}(n)$ is the sample mean, $t_{n-1,1-\alpha/2}$ is the t-test statistic, $S^2(n)$ is the sample variance and $100x\alpha$ is the confidence level expressed as a percentage.

The buffer statistics are calculated for requests before entering the buffer. Given the two element buffer, results are given for buffer empty, one request in the buffer and buffer overflow. For synchronous requests the arrival request rate and T_OPR are determined to guarantee that no synchronous requests are lost.

11 Conclusion

An inexpensive PC based FDDI network analyser based upon the DPTBS has been presented. The purpose of the network analyser was to provide a tool suitable for validating a theoretical and simulation models of the FDDI MAC protocol. The network analyser had to be capable of loading an FDDI network to in excess of 100 Mbps with synchronous and asynchronous traffic. It was also necessary to provide the user with a means of specifying the distribution of IATs for requests, the asynchronous priorities, and the ring delay (eg T_OPR).

The example run in Section 10 shows that it is possible to load the network to in excess of 100 Mbps. The traffic generators can produce either synchronous or asynchronous frames. The user has a variety of options in setting the request patterns of data and the operational parameters of the network.

The data logger can process and store all relevant network events in real time up to, and in excess of 100 Mbps. The analysis software can be used to generate traffic summaries and detailed lists of network events.

The use of the delay units provide extra flexibility in examining larger rings and determining the effect of increased ring latency on network performance.

This work has overcome the major problem with using PCs to generate network traffic which is to maintain a continuous load of network traffic at high request rates (greater than 30 Mbps). Unfortunately buffer limitations do limit the overall application of the analyser.

A more detailed set of experiments to test the general utilisation of the analyser and for the validation of the FDDI model will be the subject of further papers.

12 Acknowledgments

The authors wish to acknowledge with gratitude the assistance provided by Mr J.G. Schapel (ERL) and Mr M.L. Scholz (ERL) in the conception and development of this project. The authors are also grateful for the assistance with the DPTBS provided by Mr Stan Miller (ERL).

13 References

No.	Author	Title
1	Schapel, J.G.	"A Review of Data Communications in Combat Systems", ERL-0659-RN, 1986
2	Scholz, M.L.	"Simulation of the FDDI Network: A Progress Report" CSI Working Paper 90/01, 1990
3	Miller, S.J.	"Distributed Processing Test-Bed System: First Interim Report for the DPTBS" WSRL-TM-26/90, 1990
4	ANSI	"FDDI Token Ring Media Access Controller", X3.139-1987, 1986
5	ANSI	"FDDI Station Management" X3T9.5/84-49, Rev 5.1, 1989
6	ANSI	"FDDI Physical Media Dependent" X3T9.5/84-48, Rev 6, 1986
7	ANSI	"FDDI Physical Layer Protocol" X3T9.5/83-15, Rev 15, 1989
8	AMD	"THE SUPERNET Family for FDDI" Technical Manual Advanced Micro Devices, 1989
9	AMD	"THE SUPERNET Family for FDDI" 1989 Data Book Advanced Micro Devices, 1989
10	AMD	"SUPERNET Hardware and Software Support", Advanced Micro Devices, 1989
11	Nolan, T.	"Real Time Data Acquisition Using DMA", Dr Dobbs Journal, pp 28 January 1990
12	Johnson, R. Bhattacharyya, G.	"Statistics Principles and Methods" John Wiley & Sons, 1985

APPENDIX I

Detailed Validation

This appendix discusses the validation of the network analyser. Hardware was debugged using a Phillips PM3570 digital analyser. The programs PDEMO and FDDIMON (Ref. 10) supplied with the AMD "Fast Cards were also used to check the FDDI cards and their operation.

The analysis program, developed in ITD, produces detailed listings and summary results from the ARRIVAL and SERVICE files. The summary results are discussed in Section 10. The detailed listings are used mainly for validation. The operation of the analyser during an experimental run can be determined from the detailed listings. The detailed listings are therefore useful to validate the overall operation of the network analyser.

The following discussion is based upon the results of the experimental run discussed in Section 10. The results are for a single run with three nodes, each generating 70 Mbps. The following sections discuss firstly the detailed listing (Section I.1) and secondly the service history (Section I.2) explaining how requests were serviced at each node. The request and service times in this section are discussed in terms of clock ticks. The clock ticks are the time base used by the data logger and represent 500 ns of elapsed time. For example if an event occurred at 12292232 ticks the elapsed time is 6.1465 s (viz 12292232 ticks * 500 ns).

I.1 Detailed Listing

The results from the run are printed to the nominated data file, in this case 70KeyA.dat, "70" is the data rate (70 Mbps) and the frame key is "A" (node 1 transmitting synchronous requests). The batch size is 10 frames and the frame length is 2688 bytes. Node 2 transmits asynchronous high priority (2560 ns) and node 3 transmits asynchronous low priority (372000 ns). The last service time for synchronous requests is set to zero. Asynchronous requests use the last service time produced during the analysis of the synchronous requests to determine the end time of the analysis (see Section 9). In this case the last service time for synchronous request is set to zero to specify that all synchronous requests should be analysed. The last service time for the asynchronous requests is set to 12292232, specifying that all asynchronous requests after 12292232 (500 ns clock ticks) should be ignored.

In the following lists, the first column represents the frame key, "A", "B", "C" for nodes one, two and three respectively. The field "B = ", is the buffer contents when the request was made, and is either B = 0 or B = 1. The RT is the Request Time for the frames, the ST is the Start Time for the service of the frames, and the CS is for the Completion of Service for each of the frames. The access delay is calculated as Delay = ST - RT. All times are given in 500 ns clock ticks. The completion of service time is used to determine, in conjunction with the current buffer contents, whether a request should be serviced or queued.

Because a batch size of 10 was specified, the RT's for all frames in the batch are the same, that is the IAT for a batch is the inter-batch arrival time. The IAT for these requests can be seen to be constant, and are calculated from the batch size, frame size and required throughput for example ;

$$\text{Throughput} = \frac{\text{Batch\_Size} * \text{Frame\_Size (Bytes)} * \text{Bits\_per\_Byte}}{\text{IAT (seconds)}}$$

$$\text{IAT} = \frac{\text{Batch\_Size} * \text{Frame\_Size (Bytes)} * \text{Bits\_per\_Byte}}{\text{Throughput}}$$

$$= \frac{10 * 2688 * 8}{70*10^6}$$

$$= 0.003072 \text{ seconds}$$

$$= 6144 \text{ (500 ns clock ticks)}$$

(i) Node Key = A

Enter the data file : a:70keyA.dat
Enter the batch size : 10
Enter the frame length : 2688
Enter the last service time : 0
Do you want a listing (Y/N) : y

A B = 0, RT = 6144, ST = 7637, CS = 8067, Delay = 1493.0
A B = 0, RT = 6144, ST = 8069, CS = 8499, Delay = 1925.0
A B = 0, RT = 6144, ST = 8500, CS = 8930, Delay = 2356.0
A B = 0, RT = 6144, ST = 8932, CS = 9362, Delay = 2788.0
A B = 0, RT = 6144, ST = 9363, CS = 9793, Delay = 3219.0
A B = 0, RT = 6144, ST = 9794, CS = 10224, Delay = 3650.0
A B = 0, RT = 6144, ST = 10226, CS = 10656, Delay = 4082.0
A B = 0, RT = 6144, ST = 10657, CS = 11087, Delay = 4513.0
A B = 0, RT = 6144, ST = 11088, CS = 11518, Delay = 4944.0
A B = 0, RT = 6144, ST = 11520, CS = 11950, Delay = 5376.0

A B = 0, RT = 12288, ST = 13403, CS = 13833, Delay = 1115.0
A B = 0, RT = 12288, ST = 13834, CS = 14264, Delay = 1546.0
A B = 0, RT = 12288, ST = 14266, CS = 14696, Delay = 1978.0
A B = 0, RT = 12288, ST = 14697, CS = 15127, Delay = 2409.0
A B = 0, RT = 12288, ST = 15129, CS = 15559, Delay = 2841.0
A B = 0, RT = 12288, ST = 15560, CS = 15990, Delay = 3272.0
A B = 0, RT = 12288, ST = 15991, CS = 16421, Delay = 3703.0
A B = 0, RT = 12288, ST = 16423, CS = 16853, Delay = 4135.0
A B = 0, RT = 12288, ST = 16854, CS = 17284, Delay = 4566.0
A B = 0, RT = 12288, ST = 17285, CS = 17715, Delay = 4997.0

(ii) Node Key = B

Asynchronous high priority (2560 ns). The fields in this case are the same as those in the synchronous case above. Except in this case the batch size is 1 and the frame length is 4480 bytes, resulting in the required request rate of 70 Mbps (see above calculation).

Enter the data file : a:70keyB.dat
Enter the batch size : 1
Enter the frame length : 4480
Enter the last service time : 12292232
Do you want a listing (Y/N) : y

B B = 0, RT = 2560, ST = 2566, CS = 3282, Delay = 6.0
B B = 0, RT = 3584, ST = 4018, CS = 4734, Delay = 434.0
B B = 1, RT = 4096, ST = 4736, CS = 5452, Delay = 640.0
B B = 0, RT = 5632, ST = 6199, CS = 6915, Delay = 567.0
B B = 1, RT = 6144, ST = 6917, CS = 7633, Delay = 773.0
B B = 0, RT = 7680, ST = 11964, CS = 12680, Delay = 4284.0
B B = 1, RT = 9728, ST = 12682, CS = 13398, Delay = 2954.0
B B = 0, RT = 13824, ST = 17730, CS = 18446, Delay = 3906.0
B B = 1, RT = 15360, ST = 18448, CS = 19164, Delay = 3088.0
B B = 0, RT = 19968, ST = 23495, CS = 24211, Delay = 3527.0

(iii) Node Key = C

Asynchronous low priority (3372000 ns). The fields in this case are the same as those in the synchronous case above. Except in this case the batch size is 1 and the frame length is 4480 bytes, resulting in the required request rate of 70 Mbps (see above calculation).

Enter the data file : a:70keyC.dat
Enter the batch size : 1
Enter the frame length : 4480
Enter the last service time : 12292232
Do you want a listing (Y/N) : y

C B = 0, RT = 3072, ST = 3297, CS = 4013, Delay = 225.0
C B = 0, RT = 5120, ST = 5478, CS = 6194, Delay = 358.0
C B = 0, RT = 7168, ST = 30720, CS = 31436, Delay = 23552.0
C B = 1, RT = 8704, ST = 31438, CS = 32154, Delay = 22734.0
C B = 0, RT = 33280, ST = 42992, CS = 43708, Delay = 9712.0
C B = 1, RT = 33792, ST = 43710, CS = 44426, Delay = 9918.0
C B = 0, RT = 44544, ST = 92003, CS = 92719, Delay = 47459.0
C B = 1, RT = 45568, ST = 92721, CS = 93437, Delay = 47153.0
C B = 1, RT = 93184, ST = 159748, CS = 160464, Delay = 66564.0
C B = 1, RT = 93696, ST = 160466, CS = 161182, Delay = 66770.0

I.2 Service History

This section uses the detailed output discussed above to provide a service history of the requests on the network. At each step the request and service events are combined in chronological order.

1. At time ST=2566, the first request Node 2 (asynchronous Key=B, RT=2560) is serviced. This request completes service at time CS=3282, the delay is calculated as 2566 - 2560. As no other requests are at this node, the token is passed to node 3 (Key=C).

B B = 0, RT = 2560, ST = 2566, CS = 3282, Delay = 6.0

2. Node 3 (Key=C) has a request waiting (RT=3072), which starts service at ST=3297, and completes service at time 4013. No other requests are waiting the token is passed to node 1 (Key=A).

C B = 0, RT = 3072, ST = 3297, CS = 4013, Delay = 225.0

3. Node 1 has no requests waiting and the token is immediately passed to node 2. Node 2 has a request waiting (RT=3584) and service starts at time ST=4018, service is completed at time CS=4734.

B B = 0, RT = 3584, ST = 4018, CS = 4734, Delay = 434.0

4. Before the completion of service at node 2 a new request arrives and is buffered (B=1). Node 2 starts serving this request at time RT=4736 and completes service at time CS=5452. At time 5452 node 2 has no further requests waiting and the token is passed to node 3.

B B = 1, RT = 4096, ST = 4736, CS = 5452, Delay = 640.0

5. Node 3 begins service at time ST=5478 and completes service at time 6194 and passes the token to node 1.

C B = 0, RT = 5120, ST = 5478, CS = 6194, Delay = 358.0

6. Node 1 has no requests and passes the token directly to node 2. Node 2 has a request waiting (RT=5632) and starts serving ST=6199, completes service by CS=6915.

B B = 0, RT = 5632, ST = 6199, CS = 6915, Delay = 567.0

7. Before the completion of service at node 2 a new request arrives and is buffered (B=1). Node 2 starts serving this request at time ST=6917 and completes service at time CS=7633. At time 7633 node 2 has no further requests waiting and the token is passed to node 3.

B B = 1, RT = 6144, ST = 6917, CS = 7633, Delay = 773.0

During this rotation node 3 has run out of time and cannot send any requests. Node 3 has a T_PRI of 3372000 ns against the T_OPR of 4000000 ns, which means that on any given token rotation node 3 can only transmit if the token has been utilised for less than 628 μs (4000000 ns - 3372000 ns). The service time for node 2 frames is 358 μs (4480 bytes), therefore node 3 can only be able to transmit on those token rotations where node 2 transmits one or less frames. The token is passed directly to node 1.

8. Node 1 receives the token and starts servicing its first batch. The first frame in the batch is serviced at time ST=7637 and completes service at time CS=8067. All frames in the batch are serviced in order, and the batch completes service at time CS=11950. The token is passed to node 2.

A B = 0, RT = 6144, ST = 7637, CS = 8067, Delay = 1493.0
A B = 0, RT = 6144, ST = 8069, CS = 8499, Delay = 1925.0
A B = 0, RT = 6144, ST = 8500, CS = 8930, Delay = 2356.0
A B = 0, RT = 6144, ST = 8932, CS = 9362, Delay = 2788.0
A B = 0, RT = 6144, ST = 9363, CS = 9793, Delay = 3219.0
A B = 0, RT = 6144, ST = 9794, CS = 10224, Delay = 3650.0
A B = 0, RT = 6144, ST = 10226, CS = 10656, Delay = 4082.0
A B = 0, RT = 6144, ST = 10657, CS = 11087, Delay = 4513.0
A B = 0, RT = 6144, ST = 11088, CS = 11518, Delay = 4944.0
A B = 0, RT = 6144, ST = 11520, CS = 11950, Delay = 5376.0

9. Node 2 receives the token and starts service at time ST=11964, completes service at time CS=12680.

B B = 0, RT = 7680, ST = 11964, CS = 12680, Delay = 4284.0

10. Before the completion of service at node 2 a new request arrives and is buffered (B=1). Node 2 starts serving this request at time RT=9728 and completes service at time CS=7633. At time 13398 node 2 has no further requests waiting and the token is passed to node 3.

B B = 1, RT = 9728, ST = 12682, CS = 13398, Delay = 2954.0

11. Node 3 misses out again.

12. Node 1 starts servicing at time ST=13403 and services until CS=17715 and passes the token onto node 2.

A B = 0, RT = 12288, ST = 13403, CS = 13833, Delay = 1115.0
A B = 0, RT = 12288, ST = 13834, CS = 14264, Delay = 1546.0
A B = 0, RT = 12288, ST = 14266, CS = 14696, Delay = 1978.0
A B = 0, RT = 12288, ST = 14697, CS = 15127, Delay = 2409.0
A B = 0, RT = 12288, ST = 15129, CS = 15559, Delay = 2841.0
A B = 0, RT = 12288, ST = 15560, CS = 15990, Delay = 3272.0
A B = 0, RT = 12288, ST = 15991, CS = 16421, Delay = 3703.0
A B = 0, RT = 12288, ST = 16423, CS = 16853, Delay = 4135.0
A B = 0, RT = 12288, ST = 16854, CS = 17284, Delay = 4566.0
A B = 0, RT = 12288, ST = 17285, CS = 17715, Delay = 4997.0

13. This pattern of node 1 and node 2 services continues until the end of the run. The lack of service for node three is entirely due to its lack of TRT (time-outs). Services for node 3 become rare events (the next service time is ST=30720) resulting in significant delays and high buffer overflow probabilities.

C B = 0, RT = 7168, ST = 30720, CS = 31436, Delay = 23552.0

It is important to note that there is an inter-frame delay on batch requests which is caused by the FDDI card hardware (Ref. 8). This delay influences the maximum throughput for batch requests and if not taken into account results in a synchronous buffer overflow at high loadings. The delay is approximately 250 ns per frame in this case.

APPENDIX II

Fast Card Register Addresses

The tables in this appendix provide the addresses for the register calls required to program the FDDI card to transmit requests. Function calls are listed as they would be made in the Turbo C program.

Table II.1 FORMAC Commands

Command	Function Call
FORMAC T_PRI register	outport(0x431e)

Table II.2 DPC Commands

Command	Function Call
Request synchronous send message	outport(0x5312, 0)
Request asynchronous send message	outport(0x5314, 0)
Reset all DPC locks	outport(0x5316, 3)
Load DPC data memory register (mdru) with upper word	outport(0x530e, wordu)
Load DPC memory data register (mdrl) with lower word	outport(0x530c, wordl)
Read upper word DPC mdru	outport(0x530e, wordu)
Read Lower word DPC mdrl	outport(0x530c, wordl)

Table II.3 RBC Commands

Command	Function Call
Load RBC rpxs register	outport(0x6302, rpxs)
Load RBC rpxa register	outport(0x6304, rpxa)
Load RBC rpr register	outport(0x6306, rpr)
Load RBC mar register	outport(0x6308, addr)
Load RBC wpx register	outport(0x630a, wpx)
Load RBC wpr register	outport(0x630c, wpr)
Load RBC start of receive fifo	outport(0x6310, sar)
Load RBC end of receive fifo	outport(0x6314, ear)
Enable node processor write with increment	outport(0x631a, 0)
Enable node processor read with increment	outport(0x631c, 0)

Table II.4 Incidental Commands

Command	Function Call
Write 16 bits to buffer memory	outport(0x1300, int)
Fast command register used MUX	outport(0x0300, command)
Initialise Timer Interrupt	outport(0x3a0, 0x0)

A Fiber Distributed Data Interface (FDDI) Network Analyser

Alan Allwright, Reg Driver, Alan Wood

(DSTO-TR-0151)

DISTRIBUTION LIST

	Number of Copies
AUSTRALIA	
DEFENCE ORGANISATION	
S&T Program	
Chief Defence Scientist)	
FAS Science Policy)	1 shared copy
AS Science Industry External Relations)	
AS Science Corporate Management)	
Counsellor, Defence Science, London	Doc Control sheet
Counsellor, Defence Science, Washington	1
Senior Defence Scientific Adviser)	1 shared copy
Scientific Adviser - Policy and Command)	
Navy Scientific Adviser	1
Scientific Adviser - Army	Doc Control sheet and 1 distribution list
Air Force Scientific Adviser	1
Director Trials	1
Director, Aeronautical & Maritime Research Laboratory	1
Chief Air Operations Division	Doc Control sheet
Chief Maritime Operations Division	Doc Control sheet
Chief Weapon Systems Division	Doc Control sheet
Electronics and Surveillance Research Laboratory	
Chief Information Technology Division	1
Chief Electronic Warfare Division	Doc Control sheet
Chief Communications Division	Doc Control sheet
Chief Land, Space and Optoelectronics Division	Doc Control sheet
Chief High Frequency Radar Division	Doc Control sheet
Chief Microwave Radar Division	Doc Control sheet
Research Leader Command & Control and Intelligence Systems	1
Research Leader Military Computing Systems	1
Research Leader Command, Control and Communications	1
Executive Officer, Information Technology Division	Doc Control sheet
Head, Information Architectures Group	Doc Control sheet
Head, Information Warfare Studies Group	Doc Control sheet
Head, Software Engineering Group	Doc Control sheet
Head, Trusted Computer Systems Group	Doc Control sheet
Head, Advanced Computer Capabilities Group	Doc Control sheet

Head, Computer Systems Architecture Group	Doc Control sheet
Head, Systems Simulation and Assessment Group	Doc Control sheet
Head, Intelligence Systems Group	Doc Control sheet
Head Command Support Systems Group	1
Head, Exercise Analysis Group	Doc Control sheet
Head Information Management and Fusion Group	Doc Control sheet
Manager, Human Systems Integration Group	Doc Control sheet
Head, C3I Systems Engineering Group	1
Mr J. Schapel, Information Management & Fusion Group	1
Mr A. Allwright, C3I Systems Engineering Group	1
Mr D. O'Dea, Information Managment & Fusion Group	1
Mr A. Wood, C3I Systems Engineering Group	1
Publications and Publicity Officer, ITD	1

DSTO Library

Library Fishermens Bend	1
Library Maribyrnong	1
Library DSTOS	2
Library, MOD, Pyrmont	Doc Control sheet

Forces Executive

Director General Force Development (Sea),	Doc Control sheet
Director General Force Development (Land),	Doc Control sheet
Director General Force Development (Air),	Doc Control sheet

Navy

SO (Science), Director of Naval Warfare, Maritime Headquarters Annex, Garden Island, NSW 2000	1
Director, Naval Combat Systems Engineering	1

Army

ABCA Office, G-1-34, Russell Offices, Canberra	4

S&I Program

Defence Intelligence Organisation	1
Library, Defence Signals Directorate	Doc Control sheet

B&M Program (libraries)

OIC TRS, Defence Central Library	1
Officer in Charge, Document Exchange Centre (DEC),	1
US Defence Technical Information Center,	2
UK Defence Research Information Centre,	2
Canada Defence Scientific Information Service,	1
NZ Defence Information Centre,	1
National Library of Australia,	1

Universities and Colleges

Australian Defence Force Academy	1
Library	1
Head of Aerospace and Mechanical Engineering	1
Deakin University, Serials Section (M list)), Deakin University Library, Geelong, 3217, (Research and Technical Reports only)	1
Senior Librarian, Hargrave Library, Monash University	1

Librarian, Flinders University	1

Other Organisations

NASA (Canberra)	1
AGPS	1
State Library of South Australia	1
Parliamentary Library, South Australia	1

OUTSIDE AUSTRALIA

Abstracting and Information Organisations

INSPEC: Acquisitions Section Institution of Electrical Engineers	1
Library, Chemical Abstracts Reference Service	1
Engineering Societies Library, US	1
American Society for Metals	1
Documents Librarian, The Center for Research Libraries, US	1

Information Exchange Agreement Partners

Acquisitions Unit, Science Reference and Information Service, UK	1
Library - Exchange Desk, National Institute of Standards and Technology, US	1

SPARES 10 copies

Total number of copies: 64

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION
DOCUMENT CONTROL DATA

1. PRIVACY MARKING/CAVEAT (OF DOCUMENT)
N/A

2. TITLE

A Fiber Distributed Data Interface (FDDI) Network Analyser

3. SECURITY CLASSIFICATION (FOR UNCLASSIFIED REPORTS THAT ARE LIMITED RELEASE USE (L) NEXT TO DOCUMENT CLASSIFICATION)

Document (U)
Title (U)
Abstract (U)

4. AUTHOR(S)

Alan Allwright, Reg Driver, Alan Wood

5. CORPORATE AUTHOR

Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury SA 5108

6a. DSTO NUMBER	6b. AR NUMBER	6c. TYPE OF REPORT	7. DOCUMENT DATE
DSTO-TR-0151	AR-008-138	Technical Report	February 1995

8. FILE NUMBER	9. TASK NUMBER	10. TASK SPONSOR	11. NO. OF PAGES	12. NO. OF REFERENCES
N/A	87/226.3	Navy	42	12

13. DOWNGRADING/DELIMITING INSTRUCTIONS

14. RELEASE AUTHORITY

Chief, Information Technology Division

15. SECONDARY RELEASE STATEMENT OF THIS DOCUMENT

APPROVED FOR PUBLIC RELEASE

OVERSEAS ENQUIRIES OUTSIDE STATED LIMITATIONS SHOULD BE REFERRED THROUGH DOCUMENT EXCHANGE CENTRE, DIS NETWORK OFFICE, DEPT OF DEFENCE, CAMPBELL PARK OFFICES, CANBERRA ACT 2600

16. DELIBERATE ANNOUNCEMENT

No limitation

17. CASUAL ANNOUNCEMENT Yes

18. DEFTEST DESCRIPTORS

Fiber Distributed
Network analysers
Data Interface

The trend towards distributed Command and Control systems in Naval platforms necessitates the provision of Local Area Network performance measurement tools and techniques. A special purpose network analyser has been developed within ITD to measure the performance of a Fiber Distributed Data Interface (FDDI) network. This paper describes the architecture and operation of the network analyser.

This work has been conducted to support the Directorate of Naval Combat Systems Engineering in their Local Area Network and distributed systems analysis.